10-02-00

A

Please type a plus sign (+) inside this box [+]

PTO/SB/05 (12/97)
Approved for use through 09/30/00. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No. 042390.P8940 Total Pages 2

First Named Inventor or Application Identifier Gregory Henry

Express Mail Label No. EL431686616US

ADDRESS TO: **Assistant Commissioner for Patents**
Box Patent Application
Washington, D. C. 20231

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents.

1. X Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)

2. X Specification (Total Pages 27)
(preferred arrangement set forth below)
- Descriptive Title of the Invention
- Cross References to Related Applications
- Statement Regarding Fed sponsored R & D
- Reference to Microfiche Appendix
- Background of the Invention
- Brief Summary of the Invention
- Brief Description of the Drawings (if filed)
- Detailed Description
- Claims
- Abstract of the Disclosure

3. X Drawings(s) (35 USC 113) (Total Sheets 5)

4. X Oath or Declaration (Total Pages 5)

a. X Newly Executed (Original or Copy)

b. ___ Copy from a Prior Application (37 CFR 1.63(d))
(for Continuation/Divisional with Box 17 completed) **(Note Box 5 below)**

i. ___ DELETIONS OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).

5. _ Incorporation By Reference (useable if Box 4b is checked)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

6. _ Microfiche Computer Program (Appendix)

7. __ Nucleotide and/or Amino Acid Sequence Submission
(if applicable, all necessary)
a. ____ Computer Readable Copy

b. ____ Paper Copy (identical to computer copy)
c. ____ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. ____ Assignment Papers (cover sheet & documents(s))
9. ____ a. 37 CFR 3.73(b) Statement (where there is an assignee)
 X b. Power of Attorney
10. ____ English Translation Document (if applicable)
11. ____ a. Information Disclosure Statement (IDS)/PTO-1449
 ____ b. Copies of IDS Citations
12. ____ Preliminary Amendment
13. X Return Receipt Postcard (MPEP 503) (Should be specifically itemized)
14. ____ a. Small Entity Statement(s)
 b. Statement filed in prior application, Status still proper and desired
15. ____ Certified Copy of Priority Document(s) (if foreign priority is claimed)

16. X Other: separate sheet with title, express mail label, copy of postcard and attorney's signature

17. **If a CONTINUING APPLICATION,** check appropriate box and supply the requisite information:
___ Continuation ____ Divisional ____ Continuation-in-part (CIP)
of prior application No: _

18. **Correspondence Address**
____ Customer Number or Bar Code Label ____________________
(Insert Customer No. or Attach Bar Code Label here)
or
X Correspondence Address Below

NAME Kenneth B. Paley
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

ADDRESS 12400 Wilshire Boulevard
Seventh Floor

CITY Los Angeles STATE California ZIP CODE 90025-1026
Country U.S.A. TELEPHONE (425) 827-8600 FAX (425) 827-5644

Express Mail Label: EL431686616US

PTO/SB/17(6/99)
Approved for use through 09/30/2000. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

FEE TRANSMITTAL FOR FY 2000

TOTAL AMOUNT OF PAYMENT ($) $870.00

Complete if Known:
Application No. not yet assigned
Filing Date herewith
First Named Inventor Gregory Henry
Group Art Unit
Examiner Name
Attorney Docket No. 042390.P8940

METHOD OF PAYMENT (check one)

1. [X] **The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:**

Deposit Account Number 02-2666
Deposit Account Name

[X] **Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17**

2. X **Payment Enclosed:**
X **Check**
Money Order
Other

FEE CALCULATION

1. **BASIC FILING FEE**

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	690	201	345	Utility application filing fee	690.00
106	310	206	155	Design application filing fee	
107	480	207	240	Plant filing fee	
108	690	208	345	Reissue filing fee	
114	150	214	75	Provisional application filing fee	

SUBTOTAL (1) $ 690.00

2. **EXTRA CLAIM FEES**

			Extra Claims		Fee from below		Fee Paid
Total Claims	30	– 20** =	10	X	18	=	180.00
Independent Claims	3	– 3** =	0	X	78	=	0
Multiple Dependent						=	

****Or number previously paid, if greater; For Reissues, see below.**

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	18	203	9	Claims in excess of 20
102	78	202	39	Independent claims in excess of 3
104	260	204	130	Multiple dependent claim, if not paid
109	78	209	39	**Reissue independent claims over original patent
110	18	210	9	**Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) $ 180.00

FEE CALCULATION (continued)

Patent fees are subject to annual revisions. Small Entity payments must be supported by a small entity statement, otherwise large entity fees must be paid.
See Forms PTO/SB/09-12

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet	
139	130	139	130	Non-English specification	
147	2,520	147	2,520	For filing a request for reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for response within first month	
116	380	216	190	Extension for response within second month	
117	870	217	435	Extension for response within third month	
118	1,360	218	680	Extension for response within fourth month	
128	1,850	228	925	Extension for response within fifth month	
119	300	219	150	Notice of Appeal	
120	300	220	150	Filing a brief in support of an appeal	
121	260	221	130	Request for oral hearing	
138	1,510	138	1,510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive unavoidably abandoned application	
141	1,210	241	605	Petition to revive unintentionally abandoned application	
142	1,210	242	605	Utility issue fee (or reissue)	
143	430	243	215	Design issue fee	
144	580	244	290	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Petitions related to provisional applications	
126	240	126	240	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	
146	760	246	380	For filing a submission after final rejection (see 37 CFR 1.129(a))	
149	760	249	380	For each additional invention to be examined (see 37 CFR 1.129(a))	

Other fee (specify) ______________________ ________

Other fee (specify) ______________________ ________

SUBTOTAL (3) $ 0.00

*Reduced by Basic Filing Fee Paid

SUBMITTED BY:

Typed or Printed Name: Kenneth B. Paley

Signature [illegible] **Date** September 29, 2000

Reg. Number 38,989 **Deposit Account User ID** ________
(complete if applicable)

Our Reference: 042390.P8940 *Patent*

jc836 U.S. PTO 09/675975 09/29/00

AN INTERATIVE OPTIMIZING COMPILER
Inventors: Gregory Henry

Respectfully submitted,

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP

[illegible]

Kenneth Paley
Reg. No. 38,989

"Express Mail" mailing label number: EL431686616
Date of Deposit: September 29, 2000
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

Sharon Farnus
(Typed or printed name of person mailing paper or fee)
Sharon Farnus 9.29.00
(Signature of person mailing paper or fee) (Date signed)

09675975.092900

Serial/Patent No.: new application Filing/Issue Date: herewith
Client: Intel Corporation
Title: AN ITERATIVE OPTIMIZING COMPILER
BSTZ File No.: 042390.P8940 Atty/Secty Initials: KBP:sef
Date Mailed: September 29, 2000 Docket Due Date:

The following has been received in the U.S. Patent & Trademark Office on the date stamped hereon:

☐ Amendment/Response (____pgs.)	■ Express Mail No EL431686616US Check No 000812
☐ Appeal Brief (____pgs.) (in triplicate)	☐ ______Month(s) Extension of Time Amt: $870.00
■ Application - Utility (28 pgs., with cover and abstract)	☐ Information Disclosure Statement & PTO-1449 (____ pgs.) ☐ Check No.______ Amt:________
☐ Application - Rule 1.53(b) Continuation (____pgs.)	
☐ Application - Rule 1.53(b) Divisional (____pgs.)	☐ Issue Fee Transmittal
☐ Application - Rule 1.53(b) CIP (____pgs.)	☐ Notice of Appeal
☐ Application - Rule 1.53(d) CPA Transmittal (____pgs.)	☐ Petition for Extension of Time
☐ Application - Design (____pgs.)	☐ Petition for__________________
☐ Application - PCT (____pgs.)	■ Postcard
☐ Application - Provisional (____pgs.)	☐ Power of Attorney (____pgs.)
☐ Assignment and Cover Sheet	☐ Preliminary Amendment (____pgs.)
■ Certificate of Mailing	☐ Reply Brief (____pgs.)
■ Declaration & POA (5 pgs.) **(unsigned)**	☐ Response to Notice of Missing Parts
☐ Disclosure Docs & Orig & Copy of Inventor's Signed Letter (____pgs.)	☐ Small Entity Declaration for Indep. Inventor/Small Business
	■ Transmittal Letter, in duplicate
■ Drawings: 5 # of sheets includes 7 figures	■ Fee Transmittal, in duplicate

■ Other Copy of return postcard with certificate of mailing signed by attorney

Docket No. 042390.P8940

APPLICATION FOR UNITED STATES LETTERS OF PATENTS

FOR

AN ITERATIVE OPTIMIZING COMPILER

Inventor: Gregory Henry

Prepared by:

Blakely, Sokoloff, Taylor & Zafman
12400 Wilshire Blvd.
7th Floor
Los Angeles, CA 90025
(425) 827-8600

"Express Mail" Label No. *EL431686616US*

Date of Deposit *September 29, 2000*
I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Box Patent Application, Washington D.C. 20231

Sharon Farnus 9·29·00
Sharon E. Farnus Date

AN ITERATIVE OPTIMIZING COMPILER

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a computer implemented compiling tool, and more particularly a computer implemented method and apparatus for optimizing compiler generated object code.

Background Information

Most computer programmers write computer programs in source code using high-level languages such as C, FORTRAN, or PASCAL. While programmers may easily understand such languages, modern computers are not able to directly read and execute such languages. Thus, such computer programs must be translated into a language, known as machine language, that a specific computer can read and execute. One step in the translating process is performed by a compiler. A compiler can be thought of as a collection of interfacing algorithms, that together typically and generally translate a high-level source code program into an object code program (where it is understood that the term as used in this specification refers to any mechanism that transforms the representation of a program into another representation). The object code for execution by a computer is machine language.

A compiler generally has a front end that transforms the source code into a low level, yet largely machine independent language, called the intermediate code; and a back end that converts the intermediate code into the machine specific object code. Object code produced by conventional compilers may often be made to improve their performance in some characteristic, such as by executing faster, by minimizing bus utilization, or by requiring less memory. This improvement is called optimization. Compilers that apply code-improving transformations are called optimizing compilers.

A conventional optimizing compiler takes as input the intermediate code output by a compiler front end, and applies some static heurism to the intermediate code to hopefully produce an object code version that is more efficient than the object code would be without the transformations, without changing the meaning of the source program. These static optimization techniques include common subexpression elimination, instruction re-ordering, dummy instruction insertion, instruction substitution, loop unrolling, and loop fusion.

The problem is that prior art optimizing compilers apply only static, generic, and generally global change(s) to the code because they do not optimize for specific code variable values, and because they do not evaluate the affect of individual code changes on the performance of the object code.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be described by way of exemplary embodiments, but not limitations, illustrated in the accompanying drawings. Identical numerals indicate the same elements throughout the figures.

Figure **1** is one embodiment of a source language routine, here portrayed as a C language routine. The portrayed source language routine is an exemplar of a routine file whose object file representation can be optimized by the present invention. Figure **1** is also an example of when the target subroutine happens to call another auxiliary subroutine, not optimized by the present invention. The measuring routine portrayed in Figure **3**, and the initialization routine portrayed in Figure **4** are each written to support the exemplary optimizing of the source language routine's object file representation.

Figure **2** is one embodiment of a source language routine portrayed as a C language routine and called by the routine portrayed in Figure **1**.

Figure **3** is one embodiment of an exemplary measuring routine of the present invention that commands the measurement of a characteristic of an execution of the measuring routine, and of a compiled version of a source code routine that is a component of the measuring routine. The measuring routine is here portrayed as a C language routine that commands an execution of a machine language representation of the source routine portrayed in Figure **1** and measures execution time. The optimizing compiler of the present invention is referred to therein as the supercompiler.

Figure **4** is one embodiment of an exemplary initialization routine of the present invention that provides an initial value to the source code routine before its machine language representation execution. The optimizing compiler of the present invention is referred to therein as the supercompiler.

Figure **5** is one embodiment of a flow chart of a process of the present invention.

Figure **6** is one embodiment of an interactive menu driven by the present invention, for user selection of optimization strategies of the present invention.

Figure **7** is one embodiment of a computing system of the present invention.

.

DETAILED DESCRIPTION

In the following description, various aspects and details of the present invention will be described. However, it will be apparent to those skilled in the art that the present invention may be practiced with only some or all aspects of the present invention. For purposes of explanation, specific numbers, materials and configurations are set forth in order to provide a thorough understanding of the present invention. However, it will also be apparent to one skilled in the art that the present invention may be practiced without the specific aspects and details. In other instances, well known features are omitted or simplified in order not to obscure the present invention.

Some portions of the descriptions that follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as “processing” or “computing” or “calculating” or “determining” or “displaying” or the like, refer to the action and processes of a circuit that can include a programmed computer system, or similar electronic computing device. A computer system manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention also relates to apparatus including circuits for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may include a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium. A machine readable storage medium includes any mechanism that provides (i.e. stores and/or transmits) information in a form readable by a machine (e.g. a computer). For example, a machine-readable medium includes read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.. The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

Various operations will be described as multiple discrete steps performed in turn in a manner that is most helpful in understanding the present invention, however, the order of description should not be construed as to imply that these operations are necessarily order dependent on the order the steps are presented. Furthermore, the phrase "in one embodiment" and "an embodiment" will be used repeatedly, however the phrase does not necessarily refer to the same embodiment, although it may.

Referring to Figure **1**, an exemplary print-out of a source language routine written in the C programming language is shown. The exemplary routine portrayed in Figure **1** is a subroutine named "foo" that has five arguments, "A", "B", "C", "M", and "N", and itself calls a subroutine named samp_auxiliary_routine that has two arguments, "A" and "B", portrayed in Figure **2** as a C programming language routine. The iterative optimizer of the present invention operates on the compiled code of the target subroutine in Figure **1** by executing changes to its intermediate language representation to produce an optimized object routine. Thus, any high level language that can be compiled into an object language representation is a candidate for optimization by the method and apparatus of the present invention. The present invention in broad terms iteratively implements an optimizing change to an intermediate language representation, transforms the changed intermediate language representation into a machine language representation, selectively initializes the machine language routine, executes the machine language routine in a computing system having the architecture of the target machine, measures a user selected execution characteristic such as execution time or memory usage, and determines if a stopping criterion has been met. If the stopping criterion has not been met, the method and apparatus repeat the aforesaid process, saving the version of the most optimized routine according to the selected criterion. A preferred target routine for optimization using the method and apparatus of the present invention is a routine that may used repeatedly and in which it is desirable to have a highly optimized object code version. The resulting output of the iterative optimizer of the present invention is the fastest code the system is able to find for the target machine. The key elements of this technique are using reverse communication as a performance metric, placing instructions in main memory, and testing them iteratively, using some novel searching methods highlighted later.

Referring now to Figure **3**, a preferred embodiment exemplary timing routine, named in the portrayed embodiment "supercompiler_timer", is input in the computing system of the present invention to preferably indicate that the iterative optimizer is to measure the execution time of the timing routine. The measuring routine is preferably implemented as a C source language routine. The exemplary timing routine includes a C language mechanism to access "foo" by utilization of the C language keywords "extern void" and an argument preceded by an "*", well known to those skilled in the C programming language, as a pointer to a function defined externally to the given routine, where "void" indicates that the routine itself does not return an argument value.

The timing routine itself includes at least a call to the routine being optimized so that a machine language representation of that routine is executed when the timing routine is executed and timed. Thus, in order to time the exemplary subroutine named "foo", a user includes a call to "foo" within the timing routine. An alternative embodiment of the present invention can include a straightforward execution and measurement of each machine language representation. Yet another alternative embodiment of the present invention would require the timing routine to explicitly return a metric of performance (like the timing). In this embodiment, the timing routine preferably implemented does not explicitly itself implement a timing, but instead commands in a convenient grammar the computing system of the present invention to execute and time any code included within the timing routine. The timing itself, or any other requested figure of merit, occurs by conventional program statements and systems calls within the iterative optimizer embedded code. The timing routine here implemented provides a flexibility to time more than just a routine to be optimized, but any additional code or combinations of the routine. In the example portrayed, "foo" using as arguments "A", B", "C", "&N", "&M", and also using the arguments "A" and "B" reversed, are timed together.

This mechanism provides a convenient and flexible way for a user to request a measurement of any of a plural number of execution figures of merit, such as bus usage or memory utilization, by inputting to the iterative optimizer computing system a pertinent calling routine for the figure of merit, and the specification of what code is to be measured, where, by way of example, the routine "supercompiler_timer" for measuring an execution timeis used. The preferred characteristic to optimize is the execution time of the machine language representation.

Referring to Figure 4, an exemplary initialization routine "supercompiler_initialize()" preferably written as a C language source program routine initializes the non-globally defined variables of the object language representation by directly assigning values to the source language variables. The globally defined variables, here "N" and "M", and the data types "N", "M", "A", "B", and "C" are each defined outside supercompiler_initialize. Each variable is preferably specifically initialized. Variables not specifically initialized are assigned a default value by the iterative optimizer software of the present invention. The exemplary initialization routine includes a "supercompiler_binding" statement for binding addresses as a preferred mechanism to pass the address of any auxilary subroutines that the target subroutine being converted to machine code requires to the iterative optimizer of the present invention. The first argument in supercompiler_binding, "'samp_auxiliary_routine_'", is the name of the routine required by the routine being optimized, and the second argument, "&samp_auxiliary_routine_", is the address in memory of the additional routine. If the target subroutine "foo" does not require calling any other subroutines, then the "supercompiler_binding" is not necessary. In the preferred embodiment, the iterative optimizer will assign an instruction address for any auxilary subroutines that "foo" requires at link time. Neither the supercompiler_initialize as implemented, nor the supercompiler_timer portrayed with reference to Figure **2**, take any parameters, and therefore all the variables are global. This is an implementation decision. The exemplary supercompiler_initialize initializes "A" and "B", stores the data, and binds the object code representation by passing to the iterative optimizer of the present invention the memory addresses required for "foo". Furthermore, the initializing routine exemplary flushes cache with the "flush_cache_routine" written in the syntax of a C language program, placing the cache into a known state, so that each execution of a version of "foo" includes a common baseline cache. The "flush_cache()" routine is an example of an initialization of memory to both create a common baseline for executing iterative versions of an object language representation of the routine being optimized, and also to initialize system/cache memory to a specific state for generating a routine version optimized for a specific cache/memory state. In other embodiments of the invention, a user can directly initialize preferred memory states.

A user of the present invention inputs into a computing system the source language routine to be optimized discussed with reference to Figure **1**, the measuring routine discussed with reference to Figure **3**, and the initialization routine discussed with reference to Figure **4**. The preferred method and apparatus of the present invention is a library resident in computer memory that links with the intermediate language representation of the routine being optimized, the initialization routine, and the measuring routine. It implements a sequence of optimization changes to the intermediate language routine as shall be discussed with reference to Figure **5** according to algorithms, and executes the values. In the preferred embodiment of the present invention, the iterative optimizer software is a library and the initializing and timing routines are linked with the routine to be optimized and together they are all launched as an application.

A user attempting to optimize the target subroutine in Figure **1** would first write some high level code to measure the performance of the target subroutine (and this is what Figure **3** does) and some code to initialize the variables required by the target subroutine (and this is what Figure **4** does). Compilers in previous art do not have access to this crucial bits of information (a method of explicitly measuring the performance of a routine). This invention requires information and code similar to Figures 3 and 4 be provided so that this invention can do more effective optimization than previously possible.

Referring now to Figure **5**, the present invention includes both a method or process of compiling a source language routine, an apparatus that includes structures that perform the functions herein described, and program instructions in a machine readable medium which when executed by at least one processor cause the processor to perform the specified operations. In block **505**, a source language routine that needs optimization is read into a computing system. The computing system has stored therein an executable form of the iterative optimizer software of the present invention. The iterative optimizer software is preferably written in the C higher-level computer language. The computing system also has stored therein a compiler having a front end that generates an intermediate code translation of an input source language routine, and a back end that generates an executable code translation of the intermediate code. The preferred embodiment of the present invention includes a conventional compiler linked to the iterative optimizer of the present invention. However, it is understood that the present invention includes an embodiment having a compiler and an iterative optimizing code structured in a higher level language representation as a unitary routine.

In block **510**, a user supplied specification of initial data values of the source language routine variables is preferably read into the computing system in the form of computer software language statements. Each specified variable value is assigned to the corresponding variable before routine execution. The user may also specify initial memory assignments for the routine variables, as well as initialize the cache and memory state. The initialization routine is presented in greater detail with reference to Figure **3**. How the present invention preferably executes the machine language representation to determine the optimized code shall be disclosed presently. A most optimized representation of the object code is dependent upon the initial value of its variables. A specific example of this dependency is a variable value representing a number of loop iterations. A low number value may indicate that the most optimized machine language translation of the code should include an unrolling of the loop, while a high number value would undoubtedly indicate that the loop should not be unrolled. If a variable is not specifically initialized with a value, the iterative optimizer software of the present invention supplies a default value to the variable before execution. In the preferred embodiment, the user supplied specification comprises a C language routine, wherein the values are preferably linked to the variables by the iterative optimizer software of the present invention.

In block **515**, a user supplied measuring routine for specifying a measurable characteristic of an execution of a compiled version of the source language routine is provided to the iterative optimizer software of the present invention. The characteristic can be any measurable figure of merit of the optimized executable code such as bus utilization, memory utilization, and execution time, and is preferably the execution time of the optimized machine language representation of the source language routine. The measuring routine can specify the measurement of an execution of any combination of object language representations and specifiable discrete program steps, as presented with reference to Figure **3**. In the preferred embodiment of the present invention, the measuring routine is a C language routine that is compiled and executed under the control of the code optimizing software to time the execution time of each optimized version of the machine language representation of the source language routine.

In blocks **520** – **535**, an optimization is applied to the intermediate language code, the intermediate language code is converted to machine language, the initial values of the routine variables are assigned to the variables, the machine language code is executed, the effectiveness of the executed code is measured, and a determination is made whether a stopping criterion has been met. If the stopping criterion has not been met, blocks **520** – **535** are iterated again, wherein a new optimization is generated, the resulting code is executed in real time, and the effectiveness of the executed code is measured. Thus, a plurality of optimizations are applied, and the best one selected for the particular variable values of interest. The stopping criterion may include a specified performance gain, a specified elapsed time in finding an optimal object code version, an inability to derive an optimization attempt (which is an unlikely determination), and a relative improvement in terms of the measured effectiveness of the executed code. A new optimization is preferably selected according to at least one of a lexicographical search in which an optimization is tried each time selected from all the permutations of possible optimizations, an interactive search in which a user provides to the code optimization software, preferably through an interactive menu, the specific optimizations to try, permitting a user to interactively customize the optimization procedure, and a genetic search according to a non-linear optimization of the search domain. Furthermore, an non-optimized version of the machine language code is optionally executed and its effectiveness measured to be used as a standard in the evaluation of the optimum object routine representation.

Referring again to block **520**, a code optimizing routine makes a code optimizing change to the intermediate code representation of the source language routine. The code optimizing routine analyzes any change to substantially determine that the change does not effect the meaning of the routine, often referred to as not breaking the code. This code optimizing alteration results in an altered intermediate language routine. A novel optimizing alteration is preferably made in each iteration, as described above with reference to blocks **520 – 535**. The preferred embodiment of the present invention first attempts generic optimizations on its internal representation. These generic optimizations may include substitutions of a set of instructions, the insertion of new instructions, and the deletion of unnecessary instructions. The code optimizing routine of the present invention attempts a reordering search over possible permutations of the best effort to date to determine if key memory prefetches in different places, or other reorderings will speed up the code further. The code optimizing routine of the present invention attempts global reordering such as moving all load references to a particular variable up or down, where it is possible to do so without breaking the code. The code optimizing routine of the present invention attempts the insertion or deletion of extra instructions such as floating point stack exchange (fxch) to determine if they help or slow things down. If the target subroutine contains long strings of independent instructions (user settable, usually set at around 30), the code optimization routine inserts dependency barriers to break the search space down into segments, each one of which is independently optimized to limit the number of possible optimizing permutations in each segment.

The code optimizing routine incorporates user selectable optimization strategies. These can be interactively input into the computer system via selections menus. This enables a user to both interactively receive the intermediate and object code and the performance of the optimization process of this invention, and to respond to this information by dynamically specifying optimizing strategies based on this information. Preferred examples of interactive user inputs include: "display tasks 10-50", "produce an assembly output of the best result so far", "time and report timing on the best result so far", "move current task 29 up to position 10", "move all the loads of the array A up one position (where possible)", "move all the stores of array B down one position (where possible)". Another user selectable optimization strategy of the present invention includes a user pre-selecting the supercompiler to perform a specific search. Another, less interactive technique is to allow the optimizing routine to chose its own strategies, as long as they are in fact improvements as indicated by the user provided timing routine.

In block **525**, the altered intermediate language code is input to the compiler back end, resulting an altered machine language version of the altered intermediate code.

In block **530**, each variable initial value as described in reference to block **510** is assigned to each corresponding variable, with unassigned variables given a default value. In the preferred embodiment of the present invention, the variables are assigned their initial values by a linking procedure before execution of the machine language representation. Alternatively, the variables may be assigned their values by a parsing routine or other techniques well know to those skilled in the art.

In block **535**, the altered machine language routine is executed using the initialized values explained with reference to blocks **510** and **530**. The machine language representation is stored before execution in main memory, preferably L1 or L2, to reduce execution time so that as many distinct optimizations can be generated and tested in a time period. Furtehrmore, it is preferred that the intermediate language routine and the machine langauge routine representations remain in main memory between iterations in order to minimize database accessing time. This is significantly faster (the example in Figure 1 is easily over a thousand times faster) than the usual alternative, which is to store machine language code on disk, in object files, then go through the time consuming process of linking the object files together and running them. This process is also used to accurately measure the execution time (or other figure of merit of the compiled code) independently of extraneous and not completely consistent affects of access and search times, and to accurately compare each machine language routine. Moreover, each execution is run on a processing unit having the same memory architecture (particularly cache architecture), and processing unit architecture as the target machine, to compare the affect of the optimizations on the actual architecture for which usage is targeted. In the preferred embodiment of the present invention, block **535** is performed in a computing system having parallel processors so that a maximum quantity of distinct optimizations can be generated and tested in a period of time. Preferably, each processor can optimize a separate segment, and thus generate optimizations for each segment independently and in parallel, permitting the generation of a maximum number of solution permutations in a given time. It is also preferred that each step of the code optimizing process of the present invention be performed in the same computing system to again maximize the number of optimizations that can be generated and tested in a period of time. The characteristic of the execution is measured, preferably a timing routine for an execution time characteristic. In block **540**, if a stopping criterion has not been met, blocks **520** – **535** are iterated.

Referring to Figure **6**, an interactive menu includes menu selection items for a user selecting specific optimizations to try, as described with reference to Figure **5**. Preferred examples of interactive user inputs include: "display tasks in a range" **610**, e.g. "display tasks 10-50", "produce an assembly output of the best result so far "**615**, "time and report timing on the best result so far "**620**, "change the location of a task" **625**, e.g. "change the location of ask 29 up to position 10" **625**, "move all the loads of the array A up one position (where possible)" **630**, "move all the stores of array B down one position (where possible)" **635**.

Referring to Figure **7**, a computing system **701** is preferably identical for both compiling a source language routine representation, and executing and measuring the machine language routine representation, and it is preferred that the computing system have an identical architecture of the target machine so that machine language representation is optimized for the target machine. It is also preferred in order to test a maximum number of separate optimizations in a time period, that a system of parallel processors **701a-n** be implemented, wherein a plurality of separate processor systems, each one identical to the target system architecture (which may itself be a multi-processor architecture, generates an optimized version of the code, and executes that code, under the control of a master processor **701a**. Each parallel processing system will access the same random access memory **711**, though each processor may have its own cache. The master processor **701a** will also be coupled to a mass storage device, preferably a disk **721**, and a user interface comprising at least a display **731**, a pointing/selection device **741**, a printer **751**, and a keyboard **761**.

While certain exemplary embodiments have been described and shown in the accompanying drawings, it is to be understood that these embodiments are merely illustrative of and not restrictive of the broad invention. The present invention is not limited to the specific constructions and arrangements shown and described, and alternative embodiments will become apparent to those skilled in the art to which the present invention pertains without departing from the scope of the present invention. The scope of the present invention is defined by the appended claims rather than the foregoing description.

I claim:

1. A method of compiling a source language routine, the method comprising:

a. generating in a computer system an intermediate language routine from the source language routine;

b. specifying an initial value of each routine variable;

c. performing an optimizing change to the intermediate language routine that results in an altered intermediate language routine;

d. generating a machine language routine in a computer system from the altered intermediate language routine;

e. initializing the variables to the specified initial value;

f. executing the machine language routine from a main memory of a first computing system having the architecture of a target computer system using the initialized values;

g. measuring a characteristic of the execution; and

h. evaluating whether a stopping criterion after said executing is met and if not, repeating said performing through said measuring, saving the machine language routine having a best measured characteristic, until the stopping criterion is met.

2. The method defined in claim 1 further including before said performing:

generating a machine language routine in a computer system from the intermediate language routine;

executing the machine language routine from the main memory of the first computing system using the initialized values; and measuring a characteristic of the execution.

3. The method defined in claim 1 wherein said characteristic includes at least one of a timing wherein the best measured timing is a lowest timing, a machine language routine size, and a bus utilization metric.

4. The method defined in claim 1, further including defining a plurality of segments within the intermediate language routine, each said segment comprising consecutive intermediate language routine statements wherein no segment includes a same intermediate language routine statement, and the performing an optimizing change is performed within one of the segments.

5. The method defined in claim 1 further including determining ordering dependencies in said intermediate language routine wherein said performing an optimizing change includes maintaining the determined ordering dependencies.

6. The method defined in claim 1 wherein the optimizing change comprises one of a generic optimization, a reordering, a user selectable reordering, a user selectable global reordering, a user selectable insertion of at least one instruction in a selectable position in the intermediate language routine, and a user selectable removal of at least one instruction from a selectable position in the intermediate language routine; wherein each optimizing change does not affect the intermediate language routine integrity.

7. The method defined in claim 1 further including after the generating the machine language routine and before the executing the machine language routine, at least one user selectable optimization to the machine language routine.

8. The method defined in claim 1 wherein the optimizing changes in a sequence of a plural number of a repeated said performing resulting from the stopping criterion not met is performed according to a process that includes at least one of a non-repeating optimizing change, a user selectable optimization change sequence; and a parallel search across a plural number of processing units.

9. The method defined in claim 1 wherein the initializing further includes initializing the position of at least part of said machine language routine in the first computing system memory, and the executing includes executing the machine language using the initialized position.

10. A machine-readable medium that provides instructions, which when executed by a processor, cause said processor to perform operations comprising:

a. generating an intermediate language routine from a source language routine;

b. executing a measuring routine to define the measurement of a characteristic of an execution of a compiled representation of the routine;

c. performing an optimizing change to the intermediate language routine that results in an altered intermediate language routine;

d. generating an object language routine from the altered intermediate language routine;

e. initializing the routine variables to a user specified values;

f. measuring the characteristic of an execution of the object routine using the initialized variables and the measuring routine; and

g. evaluating whether a stopping criterion is met after said executing and if not, repeating said performing through said evaluating, saving the object language file having a best measured characteristic, until the stopping criterion is met.

11. The article defined in claim 10 wherein the operations further include before the performing:

generating an object language routine from the intermediate language routine; and measuring the characteristic of an execution of the machine language routine with the measuring routine.

12. The article defined in claim 10 wherein the characteristic includes at least one of a timing wherein the best measured timing is a lowest timing, an object code size, or a bus utilization.

13. The article defined in claim 10 wherein the operations further include defining a plurality of segments within the intermediate language routine, each said segment comprising consecutive intermediate code statements wherein no segment includes a same intermediate language routine code, and the performing an optimizing change is performed within one of the segments.

14. The article defined in claim 10 wherein the operations further include determining ordering dependencies in the intermediate language routine, wherein the performing an optimizing change includes maintaining the determined ordering dependencies.

15. The article defined in claim 10 wherein the optimizing change comprises one of a generic optimization, a reordering, a user selectable reordering, a user selectable global reordering, a user selectable insertion of at least one instruction in a selectable position in the intermediate language routine, and a user selectable removal of at least one instruction from a selectable position in the intermediate language routine; wherein each optimizing change does not affect the intermediate language file integrity.

16. The article defined in claim 10 wherein the operations further include after the generating the object language routine and before the executing, implementing at least one user selectable optimization to the object language file.

17. The article defined in claim 10 wherein the optimizing changes in a sequence of a plural number of a repeated said performing resulting from the stopping criterion not met is performed according to a process that includes at least one of a non-repeating optimizing change, a user selectable optimization change sequence; and a parallel search across a plural number of processing units.

18. The article defined in claim 10 wherein the initializing further includes initializing the position of at least a portion of the object language routine in a main memory of a computing system that performs the execution.

19. The article defined in claim 10 wherein the operations further include a user interface for reading from the user at least one of the specified values of the routine variables, and optimizing instructions wherein the performing operation includes implementing the optimizing instructions.

20. An apparatus comprising:

a. a compiler to translate a source routine into an intermediate routine, and translate an intermediate routine into an object routine;

b. a file initializer to initialize each variable of an object routine to an initial value according to a user specified value;

c. an execution characteristic measurer to measure a characteristic of an execution of the object routine;

d. an optimizer to implement an optimizing change in the intermediate routine; and

e. a computing system having an architecture of a target computing system to execute the optimized object file from a main memory of the computing system in which the variables are assigned the initial values, and to execute the execution characteristic measurer.

21. The apparatus defined in claim 20 further including an evaluator to determine if the characteristic of an execution of the optimized object routine is in accordance with a stopping criterion and if not, to repeat an execution of the optimizer, an execution of the re-optimized code, and an execution of the execution characteristic measurer, until the stopping criterion is met.

22. The apparatus defined in claim 20 wherein the optimizer further includes in a sequence of repeated executions of the optimizer, and execution of the optimized object code, at least one of a non-repeating optimizing change, and a user selectable optimization change sequence.

23. The apparatus defined in claim 20 wherein the computing system executes an object file that is not optimized in the target computing system from the main memory, that includes the initialized variables and an execution of the execution characteristic measurer.

24. The apparatus defined in claim 20 wherein the characteristic includes at least one of a timing wherein the best measured timing is the lowest timing, a machine language size, and a bus utilization.

25. The apparatus defined in claim 20 further including a code segmenter to determine a plurality of segments within the intermediate routine, each segment comprising consecutive intermediate routine statements wherein no segment includes a same intermediate language statement, and the optimizer implements a change to one of the segments.

26. The apparatus defined in claim 20 wherein the optimizing change includes a determination of ordering dependencies in the intermediate file and the optimizing change maintains the determined ordering dependencies.

27. The apparatus defined in claim 20 wherein the optimizing change includes one of a generic optimization, a reordering, a user selectable reordering, a user selectable global

reordering, a user selectable insertion of at least one instruction in a selectable position in the intermediate language routine, and a user selectable removal of at least one instruction from a selectable position in the intermediate language routine; and wherein each optimizing change does not affect the intermediate file integrity.

28. The apparatus defined in claim 20 wherein the change in the intermediate routine file includes a user selectable optimization.

29. The apparatus defined in claim 20 wherein the file initializer is further to initialize the position of at least a portion of the object routine file in the main memory, and the computing system is further to allocate the main memory according to the initialized positions of the portion of the object routine.

30. The apparatus defined in claim 20 wherein the computing system includes a plurality of processors that each have an architecture of the target computing system.

ABSTRACT

An optimizing compiler and method thereof performs a sequence of optimizing changes to an intermediate language representation of a routine, and measures an execution characteristic of each optimization, such as a timing of the machine language representation performed on an architecture similar to the target machine using a user selectable initialization state. The sequence of optimizations is selected according to a criterion that includes a lexicographic search and other methods. The pre-optimized code is also broken into segments wherein discrete optimizations are performed on each segment and measured using a user provided routine. The target routine is tested with the object code in main memory if not the cache if possible and optimizations are chosen only if they improve the target subroutine according to the user defined metric. After a stopping criterion is achieved, the most optimized code is selected.

```
subroutine foo (A,B,C,M,N)
double precision A(*), B(*), C(*), SUM1
integer M, N, I, J
call samp_auxiliary_routine(M,N)
sum1=0,d0
do I = 1, M
 sum1 = sum1 + A(I)
    do J = 1, N
          sum1 = sum1 + A((I-1)*M+J) + B((I-1)*M+J)
    enddo
    C(I) = sum1
enddo
return
end
```

Figure 1

```
samp_auxiliary_routine_(A,B)
int *A, *B;
{ *A = ARRAY_SIZE;
  *B = ARRAY_SIZE;
}
```

Figure 2

```
supercompiler_timer()
{
 extern void (*supercompiler_routine)()

 /* Example usage actually calls the target subroutine
twice, so time that:*/

 (*supercompiler_routine)(A, B, C, &N, &M);
 (*supercompiler_routine)(A, B, C, &N, &M);
}
```

Figure 3

```
#define ARRAY_SIZE 1024
int N=0
intM=0
double A[ARRAY_SIZE*ARRAY_SIZE];
double B[ARRAY_SIZE*ARRAY_SIZE];
double C[ARRAY_SIZE];

supercompiler_initialize()
{
 int i;

 /* Initialize the variables: */
 for (i = 0; i<ARRAY_SIZE*ARRAY_SIZE ; i++)
 {
  A[i] = (double) i;
  B[i] = (double) i;
 }
 for ( i = 0 ; i < ARRAY_SIZE ; i++ ) C[i] = (double) i;

 /* The target subroutine needs to call an auxiliary
routine: */

supercompiler_binding("samp_auxiliary_routine_",&samp
_auxiliary_routine_);

 /* Place the caches into a known state: clean */
 flush_cache();
}
```

Figure 4

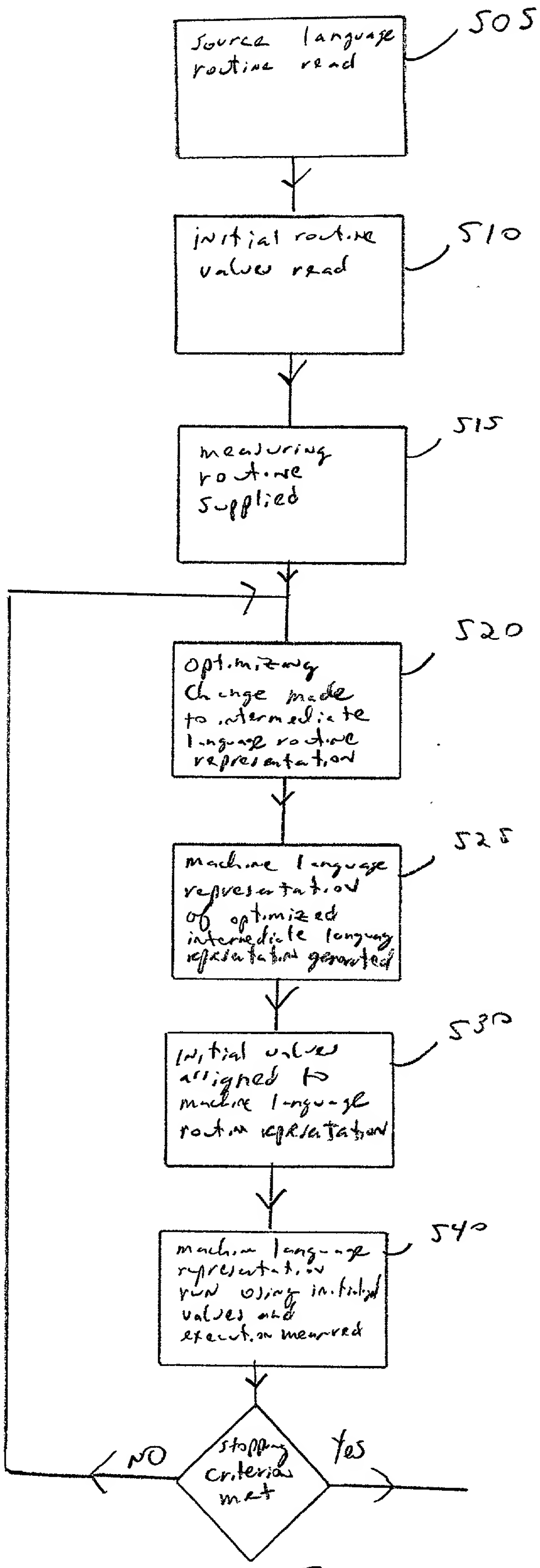
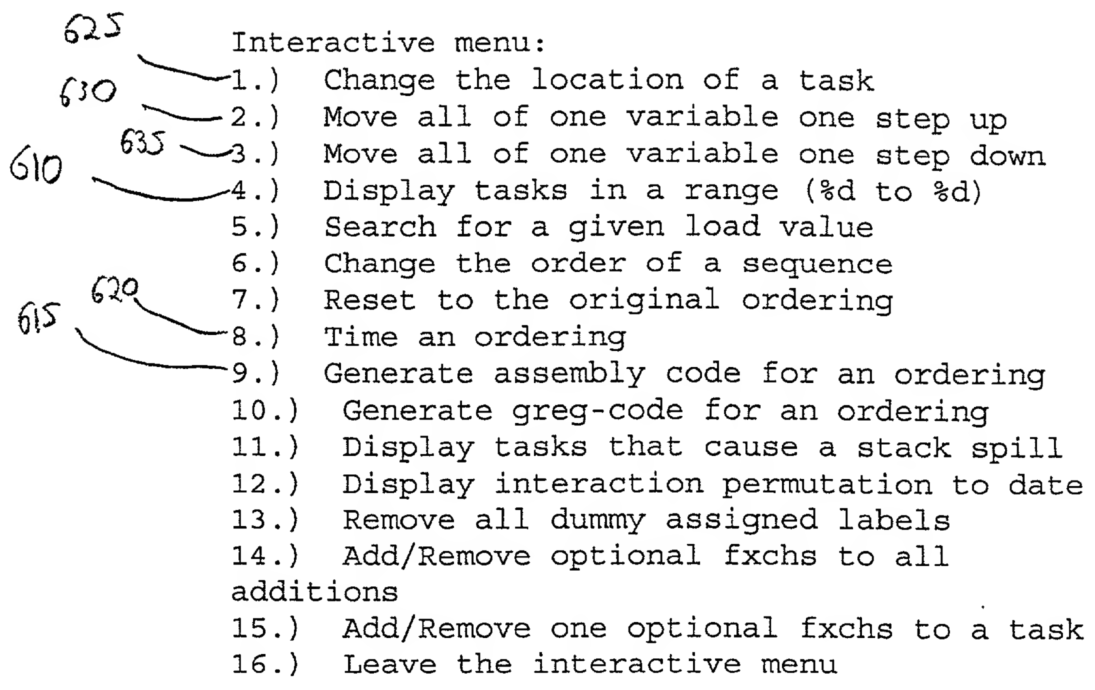

source language routine read
505
initial routine value read
510
measuring routine supplied
515
optimizing change made to intermediate language routine representation
520
machine language representation of optimized intermediate language representation generated
525
initial values aligned to machine language routine representation
530
machine language representation run using initial values and execution measured
540
stopping criteria met
NO
Yes

FIGURE 5



```
Interactive menu:
1.)  Change the location of a task
2.)  Move all of one variable one step up
3.)  Move all of one variable one step down
4.)  Display tasks in a range (%d to %d)
5.)  Search for a given load value
6.)  Change the order of a sequence
7.)  Reset to the original ordering
8.)  Time an ordering
9.)  Generate assembly code for an ordering
10.)  Generate greg-code for an ordering
11.)  Display tasks that cause a stack spill
12.)  Display interaction permutation to date
13.)  Remove all dummy assigned labels
14.)  Add/Remove optional fxchs to all
additions
15.)  Add/Remove one optional fxchs to a task
16.)  Leave the interactive menu
```

Figure 6

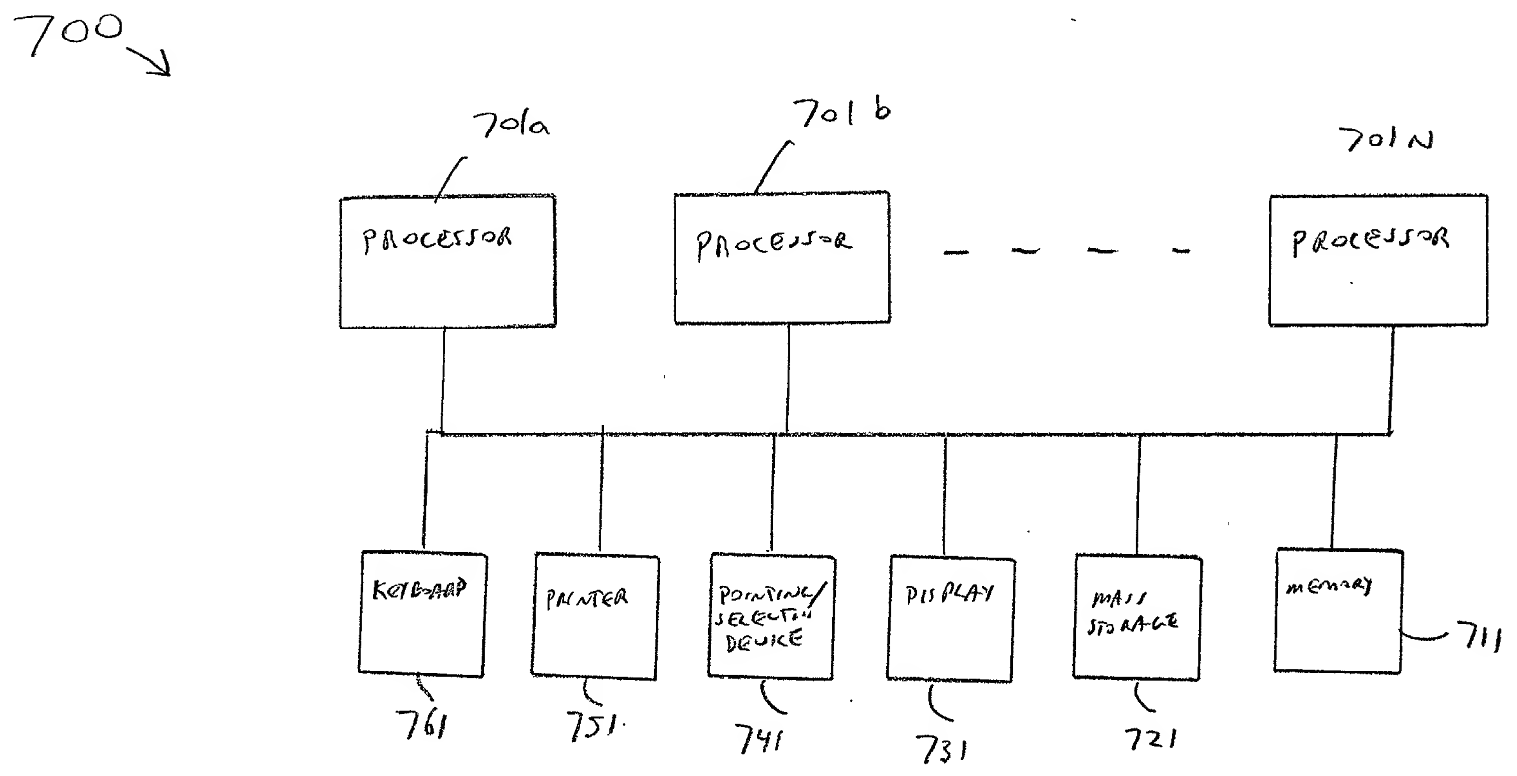

700
701a
701b
701N
PROCESSOR
PROCESSOR
PROCESSOR
KEYBOARD
PRINTER
POINTING/SELECTION DEVICE
DISPLAY
MASS STORAGE
MEMORY
761
751
741
731
721
711

FIGURE 7

Attorney's Docket No.: 042390.P8940 PATENT

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION
(FOR **INTEL CORPORATION** PATENT APPLICATIONS)

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled
AN ITERATIVE OPTIMIZING COMPILER

the specification of which

X is attached hereto.
___ was filed on ________________ as
United States Application Number ____________
or PCT International Application Number ____________
and was amended on ________________.
(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above. I do not know and do not believe that the claimed invention was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to this application, that the same was not in public use or on sale in the United States of America more than one year prior to this application, and that the invention has not been patented or made the subject of an inventor's certificate issued before the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months (for a utility patent application) or six months (for a design patent application) prior to this application.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No

I hereby claim the benefit under Title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below:

Application Number	Filing Date
Application Number	Filing Date

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

Application Number	Filing Date	Status -- patented, pending, abandoned
Application Number	Filing Date	Status -- patented, pending, abandoned

I hereby appoint the persons listed on Appendix A hereto (which is incorporated by reference and a part of this document) as my respective patent attorneys and patent agents, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Send correspondence to Kenneth Paley, BLAKELY, SOKOLOFF, TAYLOR &
(Name of Attorney or Agent)
ZAFMAN LLP, 12400 Wilshire Boulevard 7th Floor, Los Angeles, California 90025 and direct telephone calls to Kenneth Paley, (425) 827-8600.
(Name of Attorney or Agent)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor Gregory Henry

Inventor's Signature ______________________ Date ______________

Residence Hillsboro, OR Citizenship USA
(City, State) (Country)

Post Office Address 3232 NE 1st Place
Hillsboro, OR 97124

Full Name of Second/Joint Inventor ______________________

Inventor's Signature ______________________ Date ______________

Residence ______________________ Citizenship ______________
(City, State) (Country)

Post Office Address ______________________

Full Name of Third/Joint Inventor ______________________

Inventor's Signature ______________________ Date ______________

Residence ______________________ Citizenship ______________
(City, State) (Country)

Post Office Address ______________________

APPENDIX A

William E. Alford, Reg. No. 37,764; Farzad E. Amini, Reg. No. P42,261; William Thomas Babbitt, Reg. No. 39,591; Carol F. Barry, Reg. No. 41,600; Jordan Michael Becker, Reg. No. 39,602; Lisa N. Benado, Reg. No. 39,995; Bradley J. Bereznak, Reg. No. 33,474; Michael A. Bernadicou, Reg. No. 35,934; Roger W. Blakely, Jr., Reg. No. 25,831; R. Alan Burnett, Reg. No. 46,149; Gregory D. Caldwell, Reg. No. 39,926; Andrew C. Chen, Reg. No. 43,544; Thomas M. Coester, Reg. No. 39,637; Donna Jo Coningsby, Reg. No. 41,684; Florin Corie, Reg. No. 46,244; Dennis M. deGuzman, Reg. No. 41,702; Stephen M. De Klerk, Reg. No. P46,503; Michael Anthony DeSanctis, Reg. No. 39,957; Daniel M. De Vos, Reg. No. 37,813; Sanjeet Dutta, Reg. No. P46,145; Matthew C. Fagan, Reg. No. 37,542; Tarek N. Fahmi, Reg. No. 41,402; George Fountain, Reg. No. 37,374; Paramita Ghosh, Reg. No. 42,806; James Y. Go, Reg. No. 40,621; Libby N. Ho, Reg. No. P46,774; James A. Henry, Reg. No. 41,064; Willmore F. Holbrow III, Reg. No. P41,845; Sheryl Sue Holloway, Reg. No. 37,850; George W Hoover II, Reg. No. 32,992; Eric S. Hyman, Reg. No. 30,139; William W. Kidd, Reg. No. 31,772; Sang Hui Kim, Reg. No. 40,450; Walter T. Kim, Reg. No. 42,731; Eric T. King, Reg. No. 44,188; Erica W. Kuo, Reg. No. 42,775; George Brian Leavell, Reg. No. 45,436; Kurt P. Leyendecker, Reg. No. 42,799; Gordon R. Lindeen III, Reg. No. 33,192; Jan Carol Little, Reg. No. 41,181; Joseph Lutz, Reg. No. 43,765; Michael J. Mallie, Reg. No. 36,591; Andre L. Marais, under 37 C.F.R. § 10.9(b); Paul A. Mendonsa, Reg. No. 42,879; Clive D. Menezes, Reg. No. 45,493; Chun M. Ng, Reg. No. 36,878; Thien T. Nguyen, Reg. No. 43,835; Thinh V. Nguyen, Reg. No. 42,034; Dennis A. Nicholls, Reg. No. 42,036; Daniel E. Ovanezian, Reg. No. 41,236; Kenneth B. Paley, Reg. No. 38,989; Marina Portnova, Reg. No. P45,750; William F. Ryann, Reg. 44,313; James H. Salter, Reg. No. 35,668; William W. Schaal, Reg. No. 39,018; James C. Scheller, Reg. No. 31,195; Jeffrey Sam Smith, Reg. No. 39,377; Maria McCormack Sobrino, Reg. No. 31,639; Stanley W. Sokoloff, Reg. No. 25,128; Judith A. Szepesi, Reg. No. 39,393; Vincent P. Tassinari, Reg. No. 42,179; Edwin H. Taylor, Reg. No. 25,129; John F. Travis, Reg. No. 43,203; Joseph A. Twarowski, Reg. No. 42,191; Tom Van Zandt, Reg. No. 43,219; Lester J. Vincent, Reg. No. 31,460; Glenn E. Von Tersch, Reg. No. 41,364; John Patrick Ward, Reg. No. 40,216; Mark L. Watson, Reg. No. P46,322; Thomas C. Webster, Reg. No. P46,154; and Norman Zafman, Reg. No. 26,250; my patent attorneys, and Firasat Ali, Reg. No. 45,715; and Justin M. Dillon, Reg. No. 42,486; my patent agents, of BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, with offices located at 12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025, telephone (310) 207-3800, and Alan K. Aldous, Reg. No. 31,905; Edward R. Brake, Reg. No. 37,784; Ben Burge, Reg. No. 42,372; Jeffrey S. Draeger, Reg. No. 41,000; Cynthia Thomas Faatz, Reg No. 39,973; John N. Greaves, Reg. No. 40,362; Seth Z. Kalson, Reg. No. 40,670; David J. Kaplan, Reg. No. 41,105; Peter Lam, Reg. No. 44,855; Charles A. Mirho, Reg. No. 41,199; Leo V. Novakoski, Reg. No. 37,198; Thomas C. Reynolds, Reg. No. 32,488; Kenneth M. Seddon, Reg. No. 43,105; Mark Seeley, Reg. No. 32,299; Steven P. Skabrat, Reg. No. 36,279; Howard A. Skaist, Reg. No. 36,008; Gene I. Su, Reg. No. 45,140; Calvin E. Wells, Reg. No. P43,256, Raymond J. Werner, Reg. No. 34,752; Robert G. Winkle, Reg. No. 37,474; and Charles K. Young, Reg. No. 39,435; my patent attorneys, of INTEL CORPORATION; and James R. Thein, Reg. No. 31,710, my patent attorney with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

APPENDIX B

Title 37, Code of Federal Regulations, Section 1.56
Duty to Disclose Information Material to Patentability

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclosure information exists with respect to each pending claim until the claim is cancelled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is cancelled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclosure all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) Prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) The closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made or record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or

(2) It refutes, or is inconsistent with, a position the applicant takes in:

(i) Opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application;

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.